Space GUI: Python Development in Julia

Catherine Cheu

STAT 5125

December 2, 2021

(Updated December 12, 2021)

Python: Why Use It?

- Easy to learn language
- Older language with more support
- Third-party packages available

Python Implementation in Julia

- Packages used: PyCall, tkinter (Python 3.0 and later), math
- How to Call a Python Package:
  1. using PyCall
  2. Import to a package variable name:
     - [package_nickname] = pyimport("[package_name]")
  3. To call a function: [package_nickname].*function_name*(args)
     - Not doing so causes Julia to assume that the function is a Julia function instead of a Python function.

Python Coding in Julia

- Coding in Python in Julia is simple:
    - Use the following structures for Python code:
        - @pydef mutable struct *class_name* for defining classes
        - function *function_name* for defining functions
        - __init__(self) to initiate instance of classes
        - For a single line or block of pure Python code, \`py\` and \`\`\`py\`\`\` can be used, respectively.
    - Caveat: Julia syntax may still be needed for some coding (ex. true instead of TRUE which is used in Python)

tkinter Introduction

- tkinter is a Python wrapping of Tk which is available in Julia.
- Features that are used in this project:
  1. Buttons to link to functions in GUIs
  2. Labels to put on window
  3. Canvas to draw figures and animations
  4. Entry boxes to allow for user input

GUI Features

- User input: Masses of objects 1 and 2 and the distance between their centers
- Shows orbiting of a satellite around a larger mass.
  - Canvas: Blue object 1 is used as the center with red object 2 as the orbiting satellite
  - The distance between the centers of the objects can be adjusted using different inputs.
  - $G = 6.673 * 10^{-11}\ N * m^2/kg$

$$time_{orbit} = \sqrt{\frac{4\pi R^3}{GM_1}}$$

Orbits: Creation and Animation

- Creation
  - 50 points are created along the orbital path. The first point is always defined as the point directly right of the blue object.
    - The second and later points are defined as follows:
      1. An angle counter is set to zero.
      2. The counter is increased to $2\pi/50$.
      3. The point is then defined as (dist*cos(θ), dist*sin(θ)), where θ is the angle and dist is the user-defined distance between the objects' centers.
      4. The above two steps are repeated until the animation stops.

- Animation
  - When the animation starts, the satellite starts to the right of the blue object.
  - Between two points, the animation waits 40 ms before moving the satellite to the next point.
  - The above continues until:
    1. The user closes the program.
    2. The user inputs new parameters and hits the "Run" button again.

How Animation Occurs in GUI

- A button named “Run” allows the user to initiate the animation using their input.
- The canvas will move the satellite along the 50 generated points in the orbit.
- To recalculate an orbit, the user must input their desired parameters and then hit the “Run” button again. It should be noted that after the first run, the button changes into a “Recalculate” button.

Limitations

- Julia appeared to not have the same functions or equivalents as in Python.
  - Ex. __name__ variable in Python has a complicated equivalent in Julia and was not able to be used in this program.
    - As a result, this program could not implement multiple screens in the same GUI window.

Summary

- Python can be incorporated into Julia with relative ease though not all coding could be directly translated into Julia code.
- tkinter can be used to create GUIs with ease with multiple capabilities depending on user specifications.